



**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

In re application of: M. Ueda

Date: September 6, 2001

Serial No.: 09/682,188

Docket No.: JP920000046US1

Filed: August 2, 2001

Group Art Unit: 2185

FOR: Multiprocessor System, Processor Module For Use Therein, And Task Allocation  
Method In Multiprocessing

Assistant Commissioner for Patents  
Washington, D.C. 20231

**SUBMISSION OF PRIORITY DOCUMENT**

Sir:

Enclosed herewith is a certified copy of Japanese Application No. 2000-236740 filed August 4, 2000, in support of applicant's claim to priority under 35 U.S.C. 119.

Respectfully submitted,

Derek S. Jennings  
Reg. Patent Agent/Engineer  
Reg. No.: 41,473  
Tel. No.: (914) 945-2144

IBM CORPORATION  
Intellectual Property Law Dept.  
P. O. Box 218  
Yorktown Heights, N. Y. 10598



日 本 国 特 許 庁  
JAPAN PATENT OFFICE

00 046  
YOR

#5

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出 願 年 月 日

Date of Application:

2000年 8月 4日

出 願 番 号

Application Number:

特願2000-236740

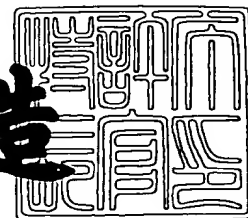
出 願 人  
Applicant(s):

インターナショナル・ビジネス・マシーンズ・コーポレーション

2001年 5月31日

特許庁長官  
Commissioner,  
Japan Patent Office

及 川 耕 造



出証番号 出証特2001-3050071

【書類名】 特許願

【整理番号】 JP9000046

【あて先】 特許庁長官殿

【国際特許分類】 G06F 12/08

【発明者】

    【住所又は居所】 滋賀県野洲郡野洲町大字市三宅 8 0 0 番地 日本アイ・  
                        ビー・エム株式会社 野洲事業所内

    【氏名】 上田 真

【特許出願人】

    【識別番号】 390009531

    【氏名又は名称】 インターナショナル・ビジネス・マシーンズ・コーポレ  
                        ーション

【代理人】

    【識別番号】 100086243

    【弁理士】

    【氏名又は名称】 坂口 博

    【電話番号】 0462-15-3318

【復代理人】

    【識別番号】 100094248

    【弁理士】

    【氏名又は名称】 楠本 高義

【選任した代理人】

    【識別番号】 100091568

    【弁理士】

    【氏名又は名称】 市位 嘉宏

【手数料の表示】

    【予納台帳番号】 012922

    【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9706050

【包括委任状番号】 9704733

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 マルチプロセッサ・システム、マルチプロセッサ・システムに用いるプロセッサ・モジュール及びマルチプロセッシングでのタスクの割り当て方法

【特許請求の範囲】

【請求項 1】 1つのキャッシュ・メモリとこのキャッシュ・メモリを共有する複数のプロセッサとを含むプロセッサ・モジュールを複数用いたマルチプロセッシングでの各プロセッサへのタスクの割り当て方法であって、

複数のプロセッサ・モジュール内のキャッシュ・メモリに記憶されている共通データへの各タスクのアクセス状況を監視するステップと、

前記アクセス状況に基づいて、同一の共通データへのアクセス回数が多いタスクを、同一プロセッサ・モジュール内のプロセッサに割り当てるステップとを含むタスクの割り当て方法。

【請求項 2】 前記アクセス状況を監視するステップが、

他のプロセッサ・モジュールのキャッシュ・メモリにも記憶されている共通データを書き換えたことによる、前記他のプロセッサ・モジュールのキャッシュ・メモリ内の共通データの無効化の発生を検出するステップと、

前記無効化を発生させたタスクの識別情報と、書き換えられたデータのアドレスと、同一タスクによる同一データでの無効化発生回数とを記憶するステップとを含み、

前記プロセッサに割り当てるステップが、

同一データでの無効化発生回数の多さに基づいて、タスクをグループ分けするステップと、

同一グループに分けられた各タスクを同一プロセッサ・モジュール内の各プロセッサに割り当てるステップとを含む請求項 1 のタスクの割り当て方法。

【請求項 3】 前記アクセス状況を監視するステップが、

複数のキャッシュ・メモリに記憶されている共通データのいずれかが書き換えられたことによる、前記共通データのうちの前記書き換えられたデータ以外のデ

ータの無効化を検出及び記憶するステップと、

前記無効化されたデータへの各タスクのアクセス状況を監視するステップとを含み、

前記プロセッサに割り当てるステップが、

同一の無効化されたデータへのアクセス回数の多さに基づいて、タスクをグループ分けするステップと、

同一グループに分けられた各タスクを同一プロセッサ・モジュール内の各プロセッサに割り当てるステップと

を含む請求項 1 のタスクの割り当て方法。

【請求項 4】 前記無効化されたデータへの各タスクのアクセス状況を監視するステップが、

無効化されたデータへのアクセスを検出するステップと、

アクセスが検出された無効化されたデータのアドレスと、この無効化されたデータへアクセスしたタスクの識別情報と、同一タスクによる同一アドレスへのアクセス回数とを記憶するステップと

を含む請求項 3 のタスクの割り当て方法。

【請求項 5】 前記無効化されたデータへのアクセスを検出するステップが

データの無効化を検出するステップと、

無効化されたデータのアドレスを記憶するステップと、

キャッシュ・ミスを検出するステップと、

キャッシュ・ミスが発生したデータ・アドレスと前記記憶された無効化されたデータのアドレスとを比較するステップと

を含む請求項 4 のタスクの割り当て方法。

【請求項 6】 前記記憶された各無効化発生回数又は各アクセス回数の合計値が所定値を超えると、前記各プロセッサへのタスクの割り当てを、マルチプロセッシングを行っているオペレーティング・システムに要求するステップを含む請求項 2、請求項 4 又は請求項 5 のタスクの割り当て方法。

【請求項 7】 1 つのキャッシュ・メモリとこのキャッシュ・メモリを共有

する複数のプロセッサとを含むプロセッサ・モジュールを複数用いたマルチプロセッサ・システムであって、

(a) 複数のプロセッサ・モジュール内のキャッシュ・メモリに記憶されている共通データへの各タスクのアクセスを検出する手段と、

アクセスが検出された共通データのアドレスと、この共通データへアクセスしたタスクの識別情報と、同一タスクによる同一データへのアクセス回数とを記憶する記憶手段と

を含むプロセッサ・モジュールと、

(b) 前記アクセス回数に基づいて、同一の共通データへのアクセス回数が多いタスクを、同一プロセッサ・モジュール内のプロセッサに割り当てる手段とを含むマルチプロセッサ・システム。

【請求項 8】 前記プロセッサ・モジュールのアクセスを検出する手段が、他のプロセッサ・モジュールのキャッシュ・メモリにも記憶されている共通データを書き換えたことによる、前記他のプロセッサ・モジュールのキャッシュ・メモリ内の共通データの無効化の発生を検出する手段と、

前記無効化を発生させたタスクの識別情報と、書き換えられたデータのアドレスと、同一タスクによる同一データでの無効化発生回数とを記憶する手段とを含む請求項 7 のマルチプロセッサ・システム。

【請求項 9】 前記プロセッサ・モジュールのアクセスを検出する手段が、複数のキャッシュ・メモリに記憶されている共通データのいずれかが書き換えられたことによる、前記共通データのうちの前記書き換えられたデータ以外のデータの無効化を検出及び記憶する手段と、

前記無効化されたデータへの各タスクのアクセスを検出する手段とを含む請求項 7 のマルチプロセッサ・システム。

【請求項 10】 前記データの無効化を検出及び記憶する手段が、データの無効化を検出する手段と、無効化されたデータのアドレスを記憶する手段とを含み、

前記無効化されたデータへの各タスクのアクセスを検出する手段が、キャッシュ・ミスを検出する手段と、キャッシュ・ミスが発生したデータ・ア

ドレスと前記記憶された無効化されたデータのアドレスとを比較する手段とを含む請求項 9 のマルチプロセッサ・システム。

【請求項 1 1】 前記記憶された各アクセス回数又は各無効化発生回数の合計値が所定値を超えると、前記各プロセッサへのタスクの割り当てを、前記割り当て手段に要求する手段をさらに含む請求項 7 又は請求項 8 のマルチプロセッサ・システム。

【請求項 1 2】 マルチプロセッサ・システムに用いる、1つのキャッシュ・メモリとこのキャッシュ・メモリを共有する複数のプロセッサとを含むプロセッサ・モジュールであって、

他のプロセッサ・モジュール内のキャッシュ・メモリにも記憶されている共通データへの各タスクのアクセスを検出する手段と、

アクセスが検出された共通データのアドレスと、この共通データへアクセスしたタスクの識別情報と、同一タスクによる同一データへのアクセス回数とを記憶する記憶手段とを含むプロセッサ・モジュール。

【請求項 1 3】 前記アクセスを検出する手段が、

他のプロセッサ・モジュールのキャッシュ・メモリにも記憶されている共通データを書き換えたことによる、前記他のプロセッサ・モジュールのキャッシュ・メモリ内の共通データの無効化の発生を検出する手段と、

前記無効化を発生させたタスクの識別情報と、書き換えられたデータのアドレスと、同一タスクによる同一データでの無効化発生回数とを記憶する手段とを含む請求項 1 2 のプロセッサ・モジュール。

【請求項 1 4】 前記アクセスを検出する手段が、

他のプロセッサ・モジュールのキャッシュ・メモリにも記憶されている共通データのうち、前記他のプロセッサ・モジュールの共通データが書き換えられたことによる、キャッシュ・メモリ内の前記共通データの無効化を検出及び記憶する手段と、

前記無効化されたデータへの各タスクのアクセスを検出する手段とを含む請求項 1 2 のプロセッサ・モジュール。



【請求項 1 5】 前記共通データの無効化を検出及び記憶する手段が、データの無効化を検出する手段と、無効化されたデータのアドレスを記憶する手段とを含み、

前記無効化されたデータへの各タスクのアクセスを検出する手段が、キャッシュ・ミスを検出する手段と、キャッシュ・ミスが発生したデータ・アドレスと前記記憶された無効化されたデータのアドレスとを比較する手段とを含む請求項 1 4 のプロセッサ・モジュール。

【請求項 1 6】 前記記憶された各アクセス回数又は各無効化発生回数の合計値が所定値を超えると、前記各プロセッサへのタスクの割り当てを、マルチプロセッシングを行っているオペレーティング・システムに要求する手段をさらに含む請求項 1 2 又は請求項 1 3 のプロセッサ・モジュール。

【発明の詳細な説明】

【0 0 0 1】

【発明の属する技術分野】

本発明は、マルチプロセッサ・システム、マルチプロセッサ・システムに用いるプロセッサ・モジュール及びマルチプロセッシングでのタスクの割り当て方法に関する。

【0 0 0 2】

【従来の技術】

M P U (Microprocessor Unit) の処理速度を高速化させる方法として、複数の M P U を使用するマルチプロセッサ・システムがある。マルチプロセッサ・システムは、複数の M P U を用いて並列処理を行う。例えば図 1 0 ( a ) に示すように、システム・メモリ 2 を共有する M P U 1 及び M P U 2 を用いてマルチプロセッサ・システムを構成する。マルチプロセッシングを行うためには、マルチプロセッサに対応している O S (Operating System) も必要になる。例えば O S に含まれるスケジューラが各 M P U へのタスク又はタスクを細分化したスレッドの割り当てを行う。図 1 1 に示すように、スケジューラが起動すると ( S 1 0 0 ) 、スケジューラは例えば優先実行順位に基づいて各タスクを各 M P U に割り当てる ( S 1 0 6 ) 。スケジューラは、タスク間の同期をとる場合等に起動される。例えば所

定時間毎にOSがスケジューラを起動する。

【0003】

図10(a)のMPU1,MPU2は、システム・メモリ2に記憶されるデータの一部が記憶されるキャッシュ・メモリA,キャッシュ・メモリBをそれぞれ備える。以下、キャッシュ・メモリを単にキャッシュとも呼ぶ。マルチプロセッサ・システムでは、システム・メモリ2を共有した状態で並列処理を行うため、各メモリ間のデータの一貫性を保つ必要がある。通常はライト・バック方式のキャッシュ制御を行うので、キャッシュA,B間のデータの一貫性を保つ必要がある。

【0004】

例えば、MPU1及びMPU2がシステム・メモリ2に記憶されているデータ(DATA\_X)を必要としている場合、図10(b)に示すように、システム・メモリ2に記憶されているDATA\_XをキャッシュAとキャッシュBにそれぞれ読み出す。MPU1とMPU2が読み出しだけを行っている間は、キャッシュA,キャッシュBの各DATA\_Xは同一のデータであり、データの一貫性は保たれている。

【0005】

しかし、例えばMPU2がDATA\_XをDATA\_Xmに更新した場合、キャッシュBに記憶されているDATA\_Xmだけが正しいデータとなり、キャッシュAに記憶されているDATA\_Xは使用してはいけない無効なデータとなる。このときデータの一貫性は保たれていない。通常、図10(c)に示すように、無効なデータDATA\_XをMPU1が使用しないように、キャッシュAに記憶されているDATA\_Xを無効化(INVALIDATE)する。

【0006】

DATA\_Xが無効化された状態で、MPU1がDATA\_Xを読み出そうとした場合、キャッシュAのDATA\_Xは無効化されているので、読み出そうとしているDATA\_XがキャッシュAに存在しないと判断される。すなわち、キャッシュ・ミスが発生する。キャッシュAにDATA\_Xが存在しないので、図10(d)に示すようにキャッシュBに記憶されているDATA\_XmをキャッシュAに読み出す。この状態では、キャッシュAとキャッシュBには、同一のデータ(DATA\_Xm)が記憶されているので、デ

ータの一貫性は保たれている。しかし、新たにDATA\_Xmが更新されると、更新されなかった方のデータの無効化、さらにはキャッシュ・ミスによるデータの読み出しが同様に起こる。

#### 【0007】

あるいは、図10(c)のDATA\_Xが無効化された状態で、MPU1がDATA\_Xを更新しようとした場合、キャッシュAのDATA\_Xは無効化されているので、キャッシュ・ミスとなる。キャッシュAにDATA\_Xmが無いので、図10(d)に示すようにキャッシュBに記憶されているDATA\_XmをキャッシュAに読み出し、任意の処理を行った後にDATA\_XmをDATA\_Xm2に更新する(図示していない)。この状態では、キャッシュAのDATA\_Xm2が正しいデータとなるので、キャッシュBのDATA\_Xmは無効化される(図示していない)。このようなデータの無効化及びキャッシュ・ミスが頻出すると、MPUの利用率が低下するので、複数のMPUを用いる意味が薄れる。以下、このようなデータの無効化及びキャッシュ・ミスが発生することを、ピンポン現象と呼ぶ。

#### 【0008】

##### 【発明が解決しようとする課題】

本発明の目的は、ピンポン現象の発生率を低下させることにある。

#### 【0009】

##### 【課題を解決するための手段】

本発明のマルチプロセッサ・システムは、(a)複数のプロセッサ・モジュール内のキャッシュ・メモリに記憶されている共通データへの各タスクのアクセスを検出する手段と、アクセスが検出された共通データのアドレスと、この共通データへアクセスしたタスクの識別情報と、同一タスクによる同一データへのアクセス回数とを記憶する記憶手段とを含むプロセッサ・モジュールと、(b)前記アクセス回数に基づいて、同一の共通データへのアクセス回数が多いタスクを、同一プロセッサ・モジュール内のプロセッサに割り当てる手段とを含む。

#### 【0010】

本発明のマルチプロセッサ・システムに用いるプロセッサ・モジュールは、他のプロセッサ・モジュール内のキャッシュ・メモリにも記憶されている共通デー

タへの各タスクのアクセスを検出する手段と、アクセスが検出された共通データのアドレスと、この共通データへアクセスしたタスクの識別情報と、同一タスクによる同一データへのアクセス回数とを記憶する記憶手段とを含む。

#### 【 0 0 1 1 】

本発明のマルチプロセッシングでのタスクの割り当て方法は、複数のプロセッサ・モジュール内のキャッシュ・メモリに記憶されている共通データへの各タスクのアクセス状況を監視するステップと、アクセス状況に基づいて、同一の共通データへのアクセス回数が多いタスクを、同一プロセッサ・モジュール内のプロセッサに割り当てるステップとを含む。

#### 【 0 0 1 2 】

##### 【発明の実施の形態】

本発明のマルチプロセッサ・システムは、1つのキャッシュとこのキャッシュを共有する複数のプロセッサとを含むプロセッサ・モジュールを複数用いる。例えば図6に示すように、キャッシュAとキャッシュAを共有するMPU1及びMPU2とを含むプロセッサ・モジュールAと、キャッシュBとキャッシュBを共有するMPU3及びMPU4とを含むプロセッサ・モジュールBとを用いてマルチプロセッサ・システムを構成する。以下、プロセッサ・モジュールを単にモジュールとも呼ぶ。

#### 【 0 0 1 3 】

図6に示すプロセッサ・モジュールを用いた場合、同一モジュール内のプロセッサ間ではキャッシュが共有されるのでピンポン現象は発生しない。例えば図7(a),(b),(c)に示すように、キャッシュAに記憶された同一データ(DATA\_X)へのMPU1とMPU2のアクセスに関してはピンポン現象は発生しない。しかし、異なるモジュールのプロセッサ間では図10(a)のマルチプロセッサ・システムと同様にピンポン現象が発生する。例えば図8(a),(b),(c)に示すように、キャッシュAとキャッシュBに記憶された同一データ(DATA\_X)へのMPU1とMPU3とのアクセスに関しては、図10(b),(c),(d)と同様にピンポン現象が発生する。ここで、図7(a)と図8(a)はシステム・メモリ2からのデータ(DATA\_X)の読み出し、図7(b)と図8(b)はMPU1によるデータ更新、図7(c),

図 8 (c) はそれぞれ M P U 2 , M P U 3 による更新後のデータ (DATA\_Xm) の読み出しを示す図である。

【 0 0 1 4 】

以下、本発明に係るマルチプロセッサ・システム、マルチプロセッサ・システムに用いるプロセッサ・モジュール及びマルチプロセッシングでのタスクの割り当て方法の実施の形態について、図面に基づいて詳しく説明する。図 6 と同様の M P U 1 , M P U 2 , キャッシュ A を含むプロセッサ・モジュール A と、 M P U 3 , M P U 4 , キャッシュ B を含むプロセッサ・モジュール B とを用いたマルチプロセッサ・システムを例にして説明する。

【 0 0 1 5 】

本発明に係るプロセッサ・モジュール A は、図 1 (a) に示すように、キャッシュ A 及びプロセッサ・モジュール B のキャッシュ B に記憶されている同一データのうちの、キャッシュ B 側のデータが書き換えられたことにより無効化されたキャッシュ A 側のデータへの各タスクのアクセスを検出するピンポン検出部 1 0 と、アクセスが検出された無効化されたデータのアドレスと、この無効化されたデータへアクセスした M P U の識別番号と、同一 M P U による同一アドレスへのアクセス回数とを含むピンポン情報を生成するピンポン情報生成部 2 0 とを含む。

【 0 0 1 6 】

ピンポン検出部 1 0 は、図 1 (b) に示すように、キャッシュ A 内のデータの無効化を検出する無効化検出器 1 2 と、無効化されたデータのアドレスを記憶する無効化アドレス記憶部 1 6 と、キャッシュ A 内のキャッシュ・ミスを検出するキャッシュ・ミス検出器 1 4 と、キャッシュ・ミスが検出されたデータのアドレスと無効化されたデータのアドレスとを比較する比較器 1 8 とを含む。無効化の検出は、例えばキャッシュ A のタグ・メモリ内の無効化されたデータ・アドレスを調べる等の方法で行うことができる。

【 0 0 1 7 】

ピンポン情報生成部 2 0 は、図 1 (b) に示すように、ピンポン情報が記憶されるピンポン情報記憶部 2 4 と、無効化されたデータへアクセスした M P U とアクセス先のデータ・アドレスをピンポン情報記憶部 2 4 に記憶し、さらに同一 M P

Uによる同一アドレスへのアクセス回数をカウントするピンポン情報更新部22とを含む。モジュールBもモジュールAと同様に、ピンポン検出部とピンポン情報生成部とを含む(図示していない)。

## 【0018】

スケジューラは、例えばタスク間の同期をとる等の目的でOSが起動するが、本発明では図2(b)に示すように、スケジューラが起動すると(S100)、ピンポン情報の読み出しを行う(S102)。さらに、MPUに割り当てられているタスクを識別した後、読み出したピンポン情報に基づいて、ピンポンの発生が最小となるように各タスクを各MPUに割り当てる(S104, S106)。具体的には、ピンポンの発生状況を解析してグループ分けし(S104)、ピンポンを頻繁に起こしたタスクどうしを同一モジュール内のMPUに割り当てる(S106)。

## 【0019】

本発明に係るマルチプロセッシングでは、図2(a)に示すように、スケジューラが起動すると(S110)タスクの割り当てを行い(S112)、キャッシュ・ミスが発生すると(S120)ピンポンを検出及び記憶し(S122)、データが無効化されると(S130)無効化されたデータ・アドレスを記憶する(S132)。

## 【0020】

次に、このようなプロセッサ・モジュールを備えたマルチプロセッサ・システムでのタスクの割り当て方法について、その作用を説明する。

## 【0021】

例えば図8(a)と同様に、MPU1とMPU2が必要とするデータがシステム・メモリ2からキャッシュAに読み出され、MPU3とMPU4が必要とするデータがシステム・メモリ2からキャッシュBに読み出される。ここで、MPU1にはタスク1が割り当てられ、MPU2, 3, 4にはタスク2, 3, 4がそれぞれ割り当てられていることとする。例えば図8(b)と同様に、キャッシュAとキャッシュBの両方に読み出されている同一データのどちらかが更新された場合、更新されなかった側のデータが無効化される。キャッシュ内のデータが無効化されると(S130)、無効化検出器12がデータの無効化を検出し、無効化されたデータのアドレスを無効化アドレス記憶部16に記憶する(S132)。

## 【 0 0 2 2 】

キャッシュ・ミスが発生すると(S 1 2 0)、キャッシュ・ミスの発生をキャッシュ・ミス検出器 1 4 が検出し、検出されたキャッシュ・ミスのデータ・アドレスと、無効化アドレス記憶部 1 6 に記憶されているアドレスとを比較器 1 8 で比較する。両者が一致すれば、無効化されたデータにアクセスが行われたことが分かる。すなわち、ピンポンが発生したことが分かる。ピンポンの発生が検出されると、ピンポンが検出された無効化されたデータのアドレスと、このアドレスにアクセスしたMPUの識別番号がピンポン情報更新部 2 2 に送られる。

## 【 0 0 2 3 】

ピンポン情報更新部 2 2 は、無効化されたデータへアクセスしたMPUの識別番号とアクセス先のデータ・アドレスとをピンポン情報記憶部 2 4 に記憶する。ここで、MPU 1 の識別番号は「1」で、MPU 2, 3, 4 の識別番号はそれぞれ「2」, 「3」, 「4」とする。ピンポン情報更新部 2 2 は、MPU及びアドレスが同一のデータが既に記憶されている場合は、その記憶データのカウンタ値を1つ増加させる。

## 【 0 0 2 4 】

ピンポン情報の一例を表 1, 2 に示す。ピンポン情報記憶部 2 4 には、ピンポンを起こしたデータのアドレス(Ping-Pong Address)と、そのアドレスにアクセスしたMPUの識別番号(MPU ID)と、同一MPUによる同一アドレスへのアクセス回数(Count)とが記憶される。表 1 はプロセッサ・モジュール A のピンポン情報であり、表 2 はプロセッサ・モジュール B のピンポン情報である。

【表 1】

Ping-Pong Address	MPU ID (Task ID)	Count
Address_A	1	100
Address_B	2	8
Address_C	2	60
Address_D	1	3

【表 2】

Ping-Pong Address	MPU ID (Task ID)	Count
Address_A	3	80
Address_A	4	20
Address_C	4	50
Address_C	3	10

【0025】

OSがスケジューラを起動すると(S100)、スケジューラはピンポン情報を読み出す(S102)。スケジューラは、MPUの識別番号(MPU ID)からそのMPUに割り当てられているタスクの識別番号(Task ID)を求め、アクセス回数(Count)に基づいて、各タスクのMPUへの割り当てを行う(S104, S106)。割り当て方法の概要を図3(a),(b)に示す。図3(a),(b)では、表1及び表2のアドレス(Ping-Pong Address)とタスク(Task ID)との緊密度をアクセス回数(Count)で表したブロック図である。現在のところ、MPU1にタスク1が割り当てられ、MPU2にタスク2が割り当てられているので、図3(b)に示すようにタスク1とタスク2を同一のグループに分けている。同様に、タスク3とタスク4を同一のグループに分けている。

【0026】

Address\_Aとの緊密度の高さは、タスク1が最も高く、次に高いのがタスク3である。Address\_Cとの緊密度の高さは、タスク2が最も高く、次に高いのがタスク4である。スケジューラは、同一アドレスへの緊密度の高いタスクどうしをグループ化する。図3(a)に示すように、Address\_Aとの緊密度の高いタスク1とタスク3とをグループ化し、Address\_Cとの緊密度の高いタスク2とタスク4とをグループ化する(S104)。

【0027】

スケジューラは、同一グループ内の各タスクを同一モジュール内の各MPUに割り当てる(S106)。例えば、プロセッサ・モジュールAのMPU1にタスク2を、MPU2にタスク4を割り当て、プロセッサ・モジュールBのMPU3にタスク1を、MPU4にタスク3を割り当てる。MPU1とMPU2間及びMP



U 3 と M P U 4 間ではピンポン現象は起こらない。図 3 ( a ) の Address\_C への Task2 と Task4 のアクセス及び Address\_A への Task1 と Task3 のアクセスではピンポンは発生しない。

#### 【 0 0 2 8 】

モジュール間の同一データへのアクセスではピンポン現象が起こる。図 3 ( b ) に示す M P U の割り当て変更前のモジュール間のアクセス回数は、

$$100 + 60 = 160 \text{ 回}$$

であるが、図 3 ( a ) に示す M P U の割り当て変更後のモジュール間のアクセス回数は、

$$20 + 10 = 30 \text{ 回}$$

であり、大幅に減少している。

#### 【 0 0 2 9 】

マルチプロセッシングでは同様の処理を繰り返し行うことが多いので、ピンポンの発生も同様な傾向で繰り返されることが多い。図 3 ( a ) に示すようにタスクの割り当てを変更することにより、ピンポンの発生は図 3 ( b ) の変更前よりも減少する可能性が高い。ピンポン現象の発生率が低下すると、各プロセッサの利用率が向上し、マルチプロセッシングの処理速度も向上する。しかも、同一データへのアクセス回数の多いタスクを同一モジュールの M P U に割り当てることで、モジュール内の各 M P U がアクセスする回数の多いデータがそのモジュール内のキャッシュに記憶されるので、キャッシュのヒット率も向上する。

#### 【 0 0 3 0 】

以上、本発明の一実施例について説明したが、本発明はその他の態様でも実施し得るものである。例えば、マルチプロセッシングに用いるプロセッサ・モジュールは 2 個に限定はされず、任意数用いることができる。プロセッサ・モジュールに含まれる M P U は 2 個に限定はされず、任意数用いることができる。

#### 【 0 0 3 1 】

プロセッサへのタスクの割り当てに限定はされず、タスクを細分化したスレッドのプロセッサへの割り当てにも本発明を用いることができる。タスクはスレッドに分けることができるので、1 つのタスクから分けられた各スレッドを各モジ

ユールの各プロセッサに割り当てる。

【 0 0 3 2 】

無効化されたデータとキャッシュ・ミスを起こしたデータのアドレスは、そのデータのアドレスを検出及び記憶することもできるが、例えばキャッシュへのデータの読み書きがキャッシュ・ライン毎に行われている場合は、そのライン毎に無効化とキャッシュ・ミスを検出及び記憶することができる。すなわち、同じキャッシュ・ラインに含まれる複数のデータを1つのグループと見なし、このグループ毎に無効化とキャッシュ・ミスを検出及び記憶する。

【 0 0 3 3 】

図4に示すように、ピンポン情報更新部32で各アクセス回数(Count)の総合計を求め、ピンポン情報内のアクセス回数の合計値が所定値を超えると、割り込みにより、タスクの割り当てをマルチプロセッシングを行っているオペレーティング・システムに要求することもできる。アクセス回数の合計値と比較する前記所定値を変更することにより、タスクの割り当てを最適化する頻度を調整することができる。このような割り込みによるタスクの割り当ての実行は、上述したスケジューラの起動の代わりに用いることもできるし、スケジューラの起動とこの割り込みとを併用することもできる。割り込みの発生は、例えば単位時間あたりのピンポン発生回数が所定値を超えた場合に発生させることもできる。

【 0 0 3 4 】

無効化されたデータへのアクセスは、書き込みと読み出しの両方を検出する以外に、書き込みアクセスのみを検出してもよいし、読み出しアクセスのみを検出してもよい。ピンポンの発生回数に基づいてタスクの割り当てを行う以外に、ピンポンが発生する前段階のデータの無効化に基づいてタスクの割り当てを行ってもよい。例えば図5に示すように、プロセッサ・モジュールBのキャッシュ・メモリBにも記憶されているキャッシュ・メモリA内の共通データを書き換えたことによる、キャッシュ・メモリB内の共通データの無効化の発生を検出する無効化検出部40、無効化を発生させたタスクの識別情報と、書き換えられたデータのアドレスと、同一タスクによる同一データでの無効化発生回数とを記憶する無効化情報記憶部42とを含ませる。無効化情報の一例を表3に示す。

【表 3】

Invalidate Address	MPU ID (Task ID)	Count
Address_A	1	100
Address_B	2	8
Address_C	2	60
Address_D	1	3

【0035】

無効化情報記憶部 42 に記憶されているアドレス (Invalidate Address) と MPU の識別番号 (MPU ID) と無効化発生回数 (Count) に基づいて、上述の実施形態と同様にタスクを MPU に割り当てることができる。ピンポンと無効化の両方を検出してもよい。さらに、データの無効化の代わりに、複数のキャッシュに記憶された同一データへの読み出しアクセスに基づいてタスクの割り当てを行うこともできる。複数のキャッシュに記憶された共通データへの読み出しアクセスの検出とデータの無効化の検出と無効化されたデータへのアクセスの検出のいずれか 1 つ又は複数を組み合わせることができる。

【0036】

複数のキャッシュに記憶されている共通データのいずれかが更新されると、更新されたデータ以外のデータは無効化され、更新されたデータは排他状態から共有状態に遷移する。この排他状態から共有状態への遷移を、無効化の代わりに検出することもできる。共有状態のデータへの各タスクのアクセス回数に基づいてタスクの割り当てを行うことができる。

【0037】

ライト・バック方式のキャッシュ制御を例にして説明したが、ライト・スルー方式のキャッシュ制御でも本発明を用いることができる。さらに DMA (Direct Memory Access) 等の MPU を介さず直接システム・メモリ 2 にアクセス可能な場合は、キャッシュ及びシステム・メモリ 2 間でデータの一貫性を保つ必要がある。例えば図 9 (a) に示ようにキャッシュ A, B にシステム・メモリ 2 の DATA\_X が読み出された状態でシステム・メモリ 2 の DATA\_X が DATA\_Xm に更新されると、図 9 (b) に示すようにキャッシュ A, B の DATA\_X を無効化する。キャッシュ B から D

ATA\_Xを読み出そうとするとキャッシュ・ミスとなり、図9(c)に示すようにシステム・メモリ2からキャッシュBへDATA\_Xmを読み出す。

【0038】

以上、本発明は特定の実施例について説明されたが、本発明はこれらに限定されるものではない。その他、本発明はその趣旨を逸脱しない範囲で当業者の知識に基づき種々なる改良、修正、変形を加えた態様で実施できるものである。

【0039】

【発明の効果】

本発明のプロセッサ・モジュールは、ピンポンを起こしたアドレスと、そのアドレスにアクセスしたタスクと、同一タスクによる同一アドレスへのアクセス回数とを監視することができる。このようなプロセッサ・モジュールを用いたマルチプロセッサ・システムは、同一データへのアクセス回数の多いタスクを同一モジュール内のプロセッサに割り当てることができる。

【0040】

本発明のマルチプロセッシングでのタスクの割り当て方法は、ピンポンを起こしたアドレスと、そのアドレスにアクセスしたタスクと、同一タスクによる同一アドレスへのアクセス回数とを監視し、同一データへのアクセス回数の多いタスクを同一モジュール内のプロセッサに割り当てて、ピンポンによるモジュール間のアクセス数が最小となるように各プロセッサへタスクを割り当てることができる。

【図面の簡単な説明】

【図1】

同図(a)は本発明に係るプロセッサ・モジュールの一構成例を示すブロック図であり、同図(b)は同図(a)に示すピンポン検出部とピンポン情報生成部の一構成例を示すブロック図である。

【図2】

同図(a)は本発明に係るマルチプロセッシングのタスクの割り当て手順の概略例を示す図であり、同図(b)は同図(a)に示すタスクの割り当て(S112)手順の詳細例を示す図である。

【図 3】

同図(a)はタスクのグループ化を最適化した例を示すブロック図であり、同図(b)はタスクのグループ化を最適化する前の例を示すブロック図である。

【図 4】

本発明に係るプロセッサ・モジュールのピンポン情報生成部の他の構成例を示すブロック図である。

【図 5】

本発明に係るプロセッサ・モジュールの他の構成例を示すブロック図である。

【図 6】

本発明に係るマルチプロセッサ・システムの基本構成を示すブロック図である。

【図 7】

同図(a),(b),(c)は図 6 に示すキャッシュ内のデータ更新の一例を示すブロック図である。

【図 8】

同図(a),(b),(c)は図 6 に示すキャッシュ内のデータの更新及び無効化の一例を示すブロック図である。

【図 9】

同図(a),(b),(c)は図 6 に示すキャッシュ内のデータの更新及び無効化の他の例を示すブロック図である。

【図 1 0】

同図(a)は従来のマルチプロセッサの一構成例を示すブロック図であり、同図(b),(c),(d)はキャッシュ内のデータの更新及び無効化の一例を示すブロック図である。

【図 1 1】

従来のマルチプロセッシングのタスクの割り当て手順の一例を示す図である。

【符号の説明】

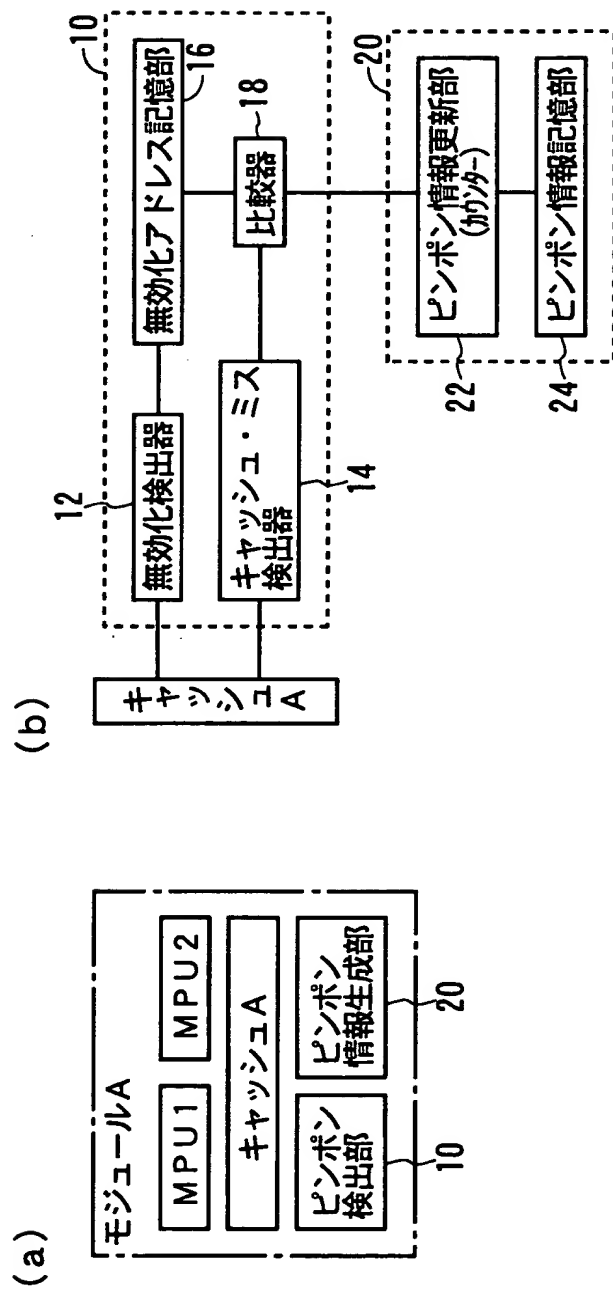
2 : システム・メモリ

1 0 : ピンポン検出部

- 1 2 : 無効化検出器
- 1 4 : キャッシュ・ミス検出器
- 1 6 : 無効化アドレス記憶部
- 1 8 : 比較器
- 2 0 , 3 0 : ピンポン情報生成部
- 2 2 , 3 2 : ピンポン情報更新部
- 2 4 : ピンポン情報記憶部
- 4 0 : 無効化検出部
- 4 2 : 無効化情報記憶部

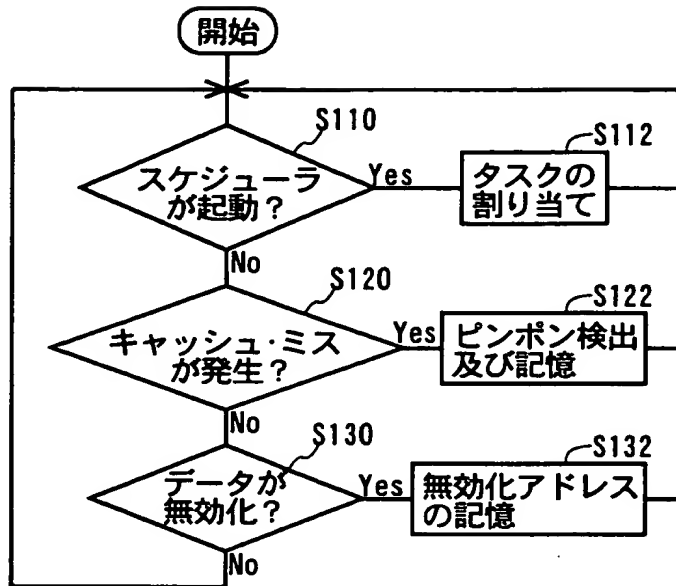
【書類名】 図面

【図 1】

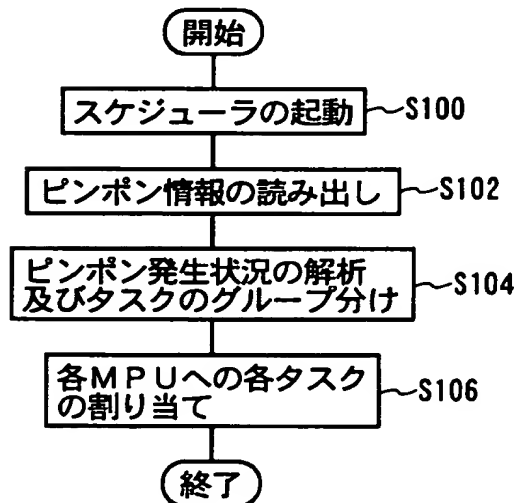


【図 2】

(a)



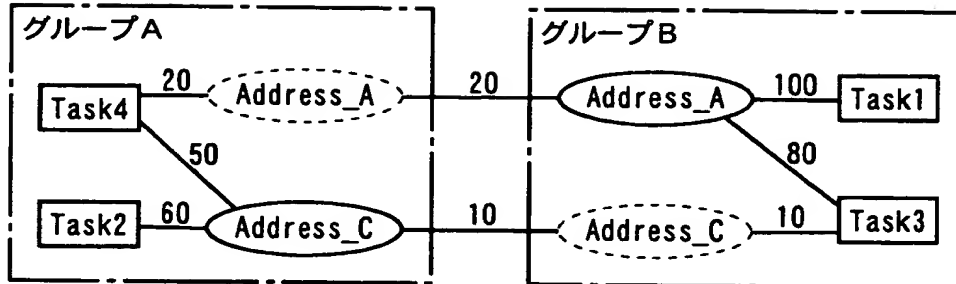
(b)



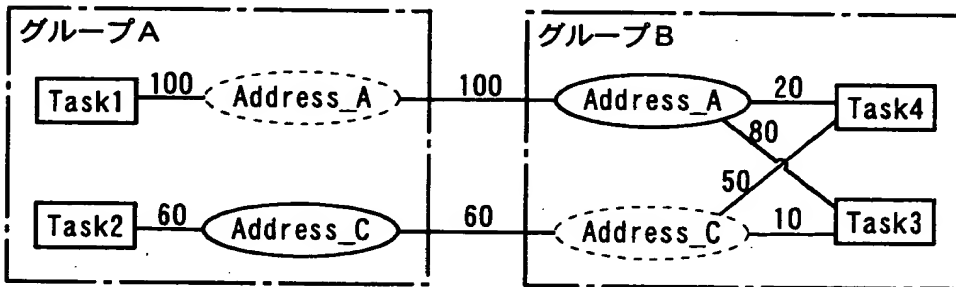


【図 3】

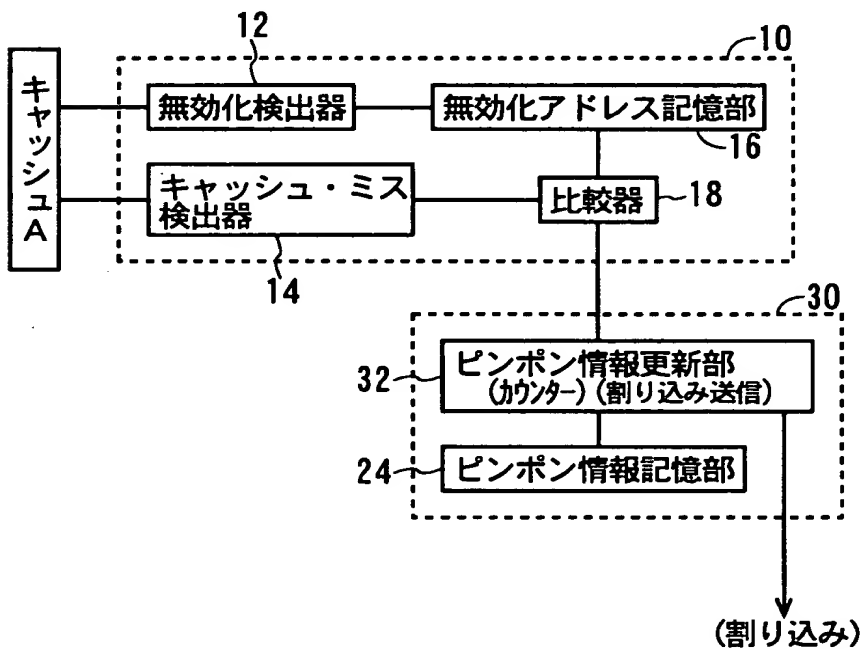
(a)



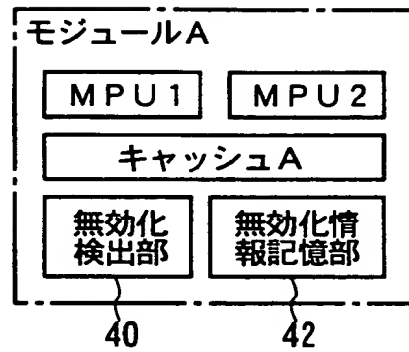
(b)



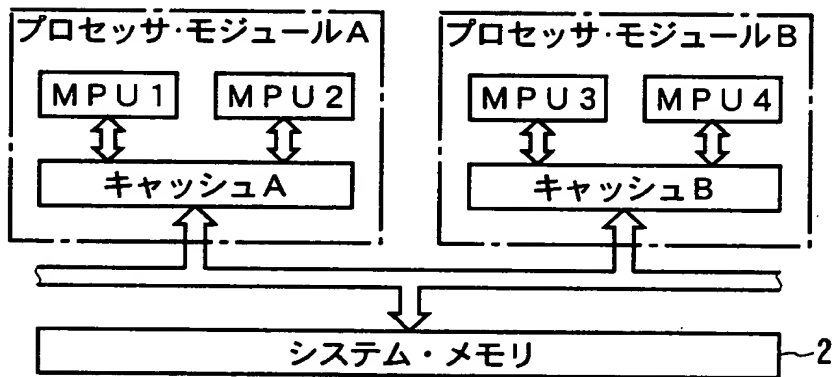
【図 4】



【図 5】

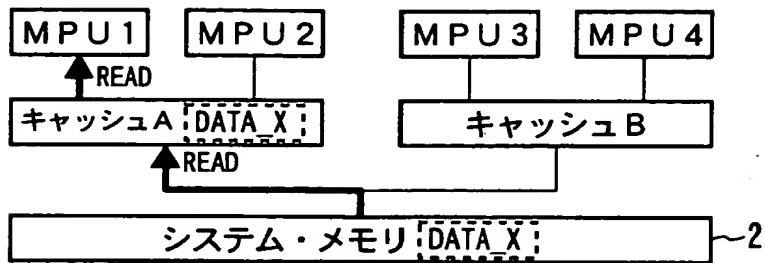


【図 6】

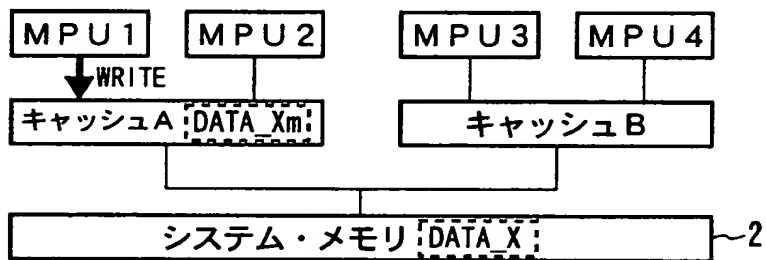


【図 7】

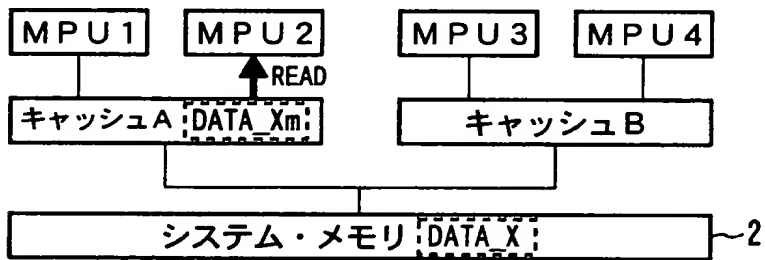
(a)



(b)

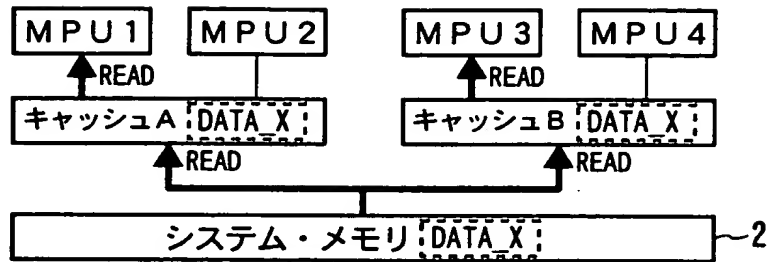


(c)

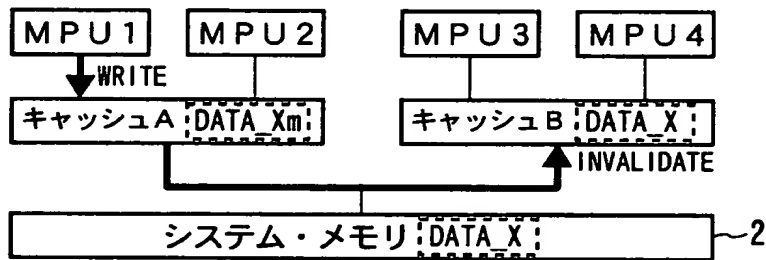


【図 8】

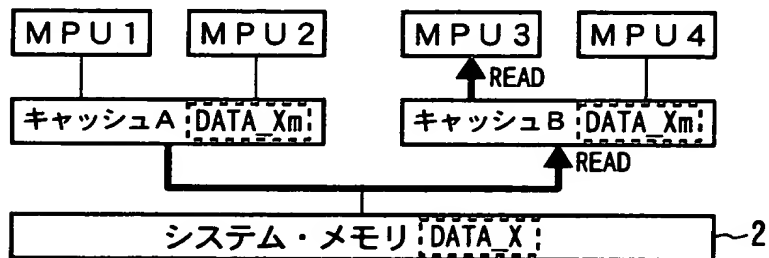
(a)



(b)

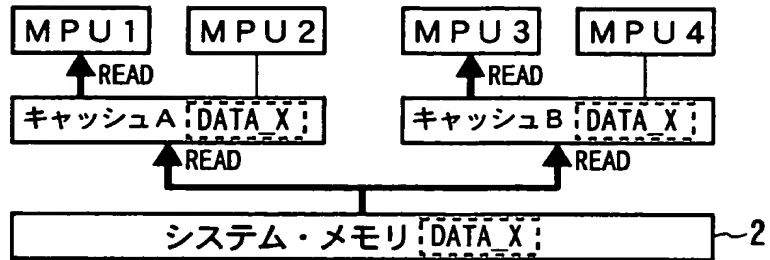


(c)

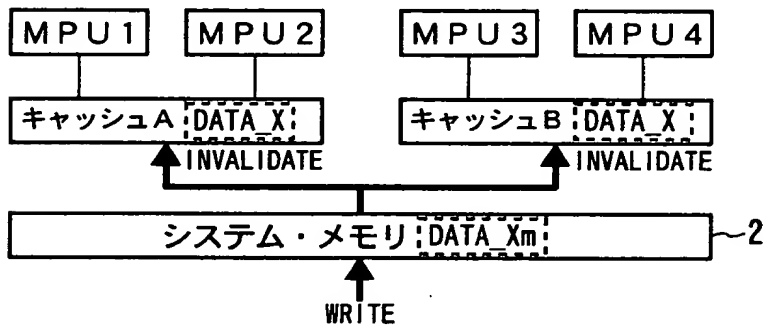


【図9】

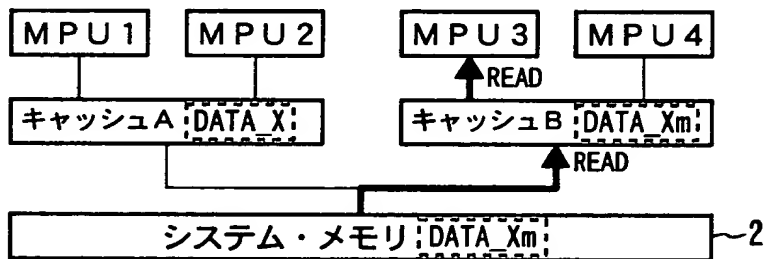
(a)



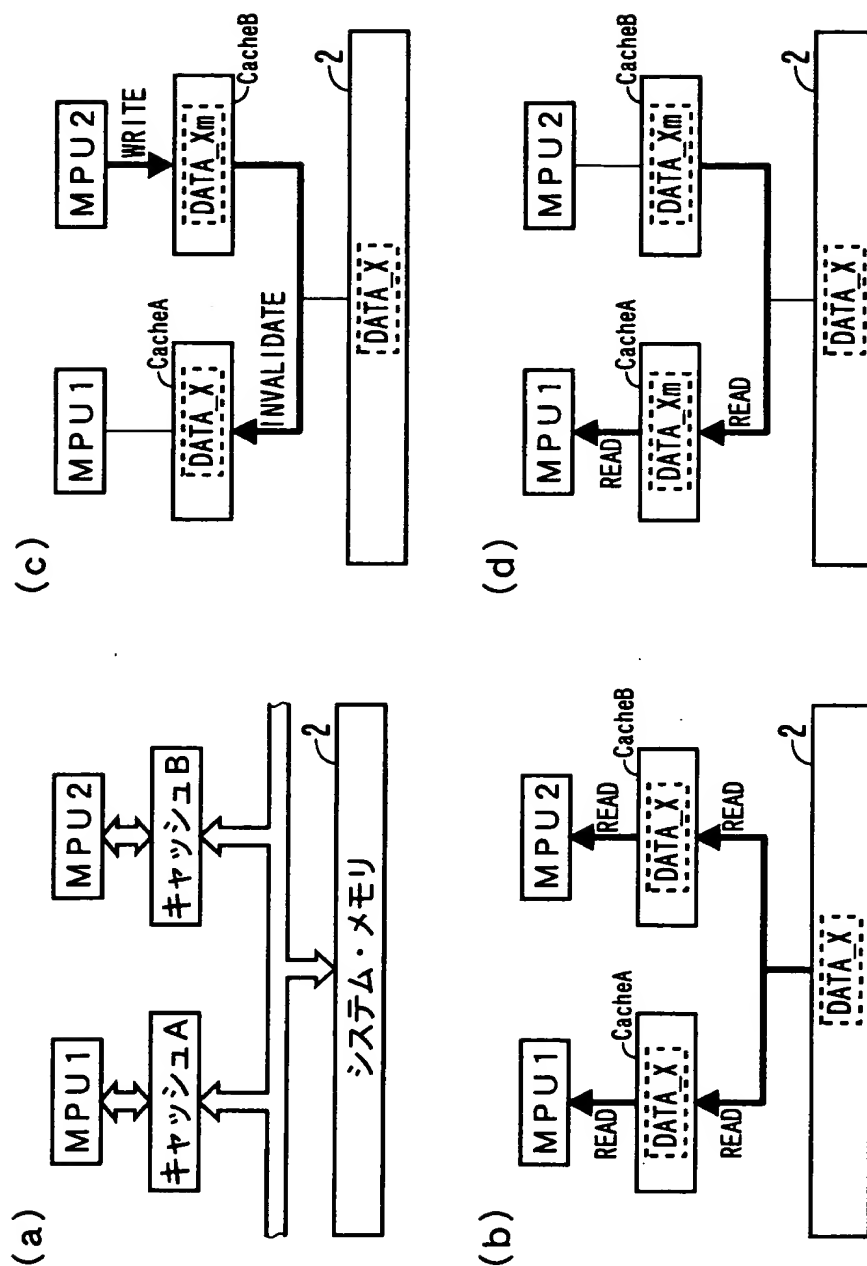
(b)



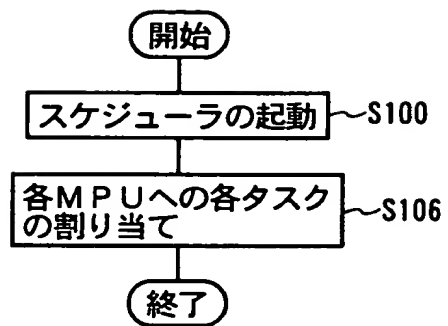
(c)



【図 10】



【図 1 1】



【書類名】 要約書

【要約】

【課題】 ピンポン現象の発生率を低下させる。

【解決手段】 他のプロセッサ・モジュール内のキャッシュ・メモリにも記憶されている共通データへの各タスクのアクセスを検出する手段 1 0 と、アクセスが検出された共通データのアドレスと、この共通データへアクセスしたタスクの識別情報と、同一タスクによる同一データへのアクセス回数とを記憶する記憶手段 2 0 とを含むプロセッサ・モジュールを複数用い、記憶手段 2 0 に記憶されたアドレスとタスクの識別情報とアクセス回数とに基づいて、同一アドレスへのアクセス回数の多いタスクを同一プロセッサ・モジュール内のプロセッサに割り当てる。

【選択図】 図 1 ( a )



認定・付加情報

特許出願の番号	特願2000-236740
受付番号	50000994394
書類名	特許願
担当官	塩崎 博子 1606
作成日	平成12年 9月21日

<認定情報・付加情報>

【提出日】	平成12年 8月 4日
【特許出願人】	
【識別番号】	390009531
【住所又は居所】	アメリカ合衆国10504、ニューヨーク州 アーモンク (番地なし)
【氏名又は名称】	インターナショナル・ビジネス・マシーンズ・コーポレーション
【代理人】	
【識別番号】	100086243
【住所又は居所】	神奈川県大和市下鶴間1623番地14 日本アイ・ビー・エム株式会社 大和事業所内
【氏名又は名称】	坂口 博
【復代理人】	申請人
【識別番号】	100094248
【住所又は居所】	滋賀県大津市栗津町4番7号 近江鉄道ビル5F 楠本特許事務所
【氏名又は名称】	楠本 高義
【選任した代理人】	
【識別番号】	100091568
【住所又は居所】	神奈川県大和市下鶴間1623番地14 日本アイ・ビー・エム株式会社 大和事業所内
【氏名又は名称】	市位 嘉宏

出 願 人 履 歴 情 報

識別番号 [390009531]

1. 変更年月日 2000年 5月16日

[変更理由] 名称変更

住 所 アメリカ合衆国10504、ニューヨーク州 アーモンク (番地なし)

氏 名 インターナショナル・ビジネス・マシーンズ・コーポレーション